Learning Tree Structure in Multi-Task Learning

Lei Han
Department of Computer Science
Hong Kong Baptist University
leihan@comp.hkbu.edu.hk

Yu Zhang[*]
Department of Computer Science
Hong Kong Baptist University
yuzhang@comp.hkbu.edu.hk

ABSTRACT

In multi-task learning (MTL), multiple related tasks are learned jointly by sharing information according to task relations. One promising approach is to utilize the given tree structure, which describes the hierarchical relations among tasks, to learn model parameters under the regularization framework. However, such a priori information is rarely available in most applications. To the best of our knowledge, there is no work to learn the tree structure among tasks and model parameters simultaneously under the regularization framework and in this paper, we develop a TAsk Tree (TAT) model for MTL to achieve this. By specifying the number of layers in the tree as H, the TAT method decomposes the parameter matrix into H component matrices, each of which corresponds to the model parameters in each layer of the tree. In order to learn the tree structure, we devise sequential constraints to make the distance between the parameters in the component matrices corresponding to each pair of tasks decrease over layers, and hence the component parameters will keep fused until the topmost layer, once they become fused in a layer. Moreover, to make the component parameters have chance to fuse in different layers, we develop a structural sparsity regularizer, which is the sum of the ℓ_2 norm on the pairwise difference among the component parameters, to learn layer-specific task structure. In order to solve the resulting non-convex objective function, we use the general iterative shrinkage and thresholding (GIST) method. By using the alternating direction method of multipliers (ADMM) method, we decompose the proximal problem in the GIST method into three independent subproblems, where a key subproblem with the sequential constraints has an efficient solution as the other two subproblems do. We also provide some theoretical analysis for the TAT model. Experiments on both synthetic and real-world datasets show the effectiveness of the TAT model.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

Multi-Task Learning, Learning Tree Structure

1. INTRODUCTION

Multi-task learning (MTL) [2] seeks to improve the generalization performance of multiple learning tasks by sharing information according to task relations. MTL has been applied in a wide range of applications including medical risk evaluation, image annotation, speech recognition, disease progression predication and so on.

From the perspective of information sharing, MTL models can be classified into two categories based on task relations. The methods in the first category utilize some domain knowledge on the task relations to devise learning models. For example, if tasks are known to be positively correlated to each other, this knowledge can be encoded into some novel regularizers [7, 6]. Moreover, some domain knowledge which specifies the tree structure among tasks has been utilized in [17, 9, 13]. However, such knowledge is rarely available in most MTL applications, which limits the use of those approaches. On the other hand, MTL methods in the second category can identify the task relations and learn the model parameters from data simultaneously. Those methods can provide insights for the problems under investigation based on the task relations revealed without any domain knowledge and hence they are the main focus of our work.

Many algorithms belonging to the second category have been proposed and shown good performance in a large number of applications. For example, low-rank structure based methods in [1, 3] can learn model parameters sharing a low dimensional subspace for multiple tasks, probabilistic MTL models [27, 25] place probabilistic priors on multiple tasks to learn the model parameters as well as identifying the task relations, regularized methods [26, 24] can reveal pairwise task relations in term of a matrix, task grouping methods [14, 16, 18] can detect the underlying task groups, robust MTL model proposed in [4] can identify the existence of outlier tasks, and some hierarchical models [15, 9, 21, 28, 12] organize the task structure into hierarchies. All the works in the second category are capable of learning some specific task structure when the domain knowledge is absent.

As discussed previously, the use of the given tree structure, which defines the task relations, has been investigated by some methods [17, 9, 13] in the first category and it leads to significant improvement in the performance. Even though learning the tree structure is claimed by those works as one promising future direction, due to the difficulty of defining and modeling the tree structure among tasks, we are not aware of any existing work that can learn the model parameters and tree structure from data simultaneously under the regularization framework. Moreover, there is even no formal definition for the tree structure in the MTL regime.

[*] Both authors contributed equally.

KDD'15 August 11 – 14, 2015, Sydney, NSW, Australia
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783393.

Table 1: Notations

Notation	Description
$\mathbf{x} \in \mathbb{R}^m$	A vector $\mathbf{x}$ with length m.
$\mathbf{X} \in \mathbb{R}^{d\times m}$	A matrix $\mathbf{X}$ with size $d \times m$.
$\mathbf{x}^i, \mathbf{x}_j, x_j^i$	The ith row, jth column, and (i, j)th element of a matrix $\mathbf{X}$.
$\|\cdot\|_q$	The ℓ_q norm defined on any vector $\mathbf{x} \in \mathbb{R}^m$, i.e., $\|\mathbf{x}\|_q = (\sum_{i=1}^m |x_i|^q)^{\frac{1}{q}}$.
$\|\cdot\|_{p,q}$	The $\ell_{p,q}$ norm defined on a matrix $\mathbf{X} \in \mathbb{R}^{d\times m}$, i.e., $\|\mathbf{X}\|_{p,q} = \left(\sum_{j=1}^d (\sum_{i=1}^m |x_j^i|^q)^{\frac{p}{q}}\right)^{\frac{1}{p}}$.
$\|\cdot\|_F$	The matrix Frobenius norm.
$\langle\cdot,\cdot\rangle$	The inner product between matrices/vectors.
$\mathrm{tr}(\cdot)$	The trace of a square matrix.
$\mathbf{I}$	An identity matrix with appropriate size.
$\mathbb{N}_m$	A set of integers $\{1, \cdots, m\}$.
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2.
$(a_1, \cdots, a_n)$	A sequence.
$(a, \cdots, a)\|_n$	A sequence with n identical elements a.
$\bowtie$	The concatenate operator between two sequences, i.e., $A \bowtie B = (A, B)$ for sequences A and B.

In this paper, we aim to fill this gap. Firstly we formally define the *task tree* structure for MTL. Based on the task tree, we develop a TAsk Tree (TAT) method for MTL to learn the underlying task tree and the model parameters simultaneously. By specifying the number of layers in the tree or equivalently the height of the tree as H, the TAT method decomposes the parameter matrix into H component matrices, each of which corresponds to the parameters in a layer of the tree. In order to learn the tree structure among tasks, we devise sequential constraints each of which makes the distance between the component parameters in the component matrices for a pair of tasks decrease over layers and as a consequence, the corresponding component parameters will keep fused until the topmost layer once they become fused in a layer. Moreover, to make the component parameters have chance to fuse together in some layers, a structural sparsity regularizer, which is defined as the sum of the ℓ_2 norm on the pairwise difference among the component parameters, is used to learn layer-specific task structure. Learning the parameters in the proposed TAT model is very challenging due to the complex fused regularizer and the sequential constraints which are non-convex. To solve the resulting objective function, we propose to use the general iterative shrinkage and thresholding (GIST) method [8], which needs to solve a proximal problem. With the help of the alternating direction method of multipliers (ADMM), we can decompose the proximal problem into three independent subproblems, where a key subproblem with the sequential constraint has an efficient solution as the other two subproblems do. Moreover, we provide theoretical analysis for the TAT model. Experiments on both synthetic and real-world datasets show that the TAT model has competitive performance over state-of-the-art MTL methods and additionally it can provide meaningful task tree structure to demonstrate the interpretability.

2. THE TAT MODEL

For clear presentation, we list the frequently used notations in Table 1. Suppose we have m learning tasks in a d-dimensional space. The training data for the ith task is denoted by $(\mathbf{X}_i, \mathbf{y}_i)$, where $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$ is the data matrix with n_i training samples stored in the rows, and $\mathbf{y}_i \in \mathbb{R}^{n_i}$ is a vector of labels for the n_i training samples in $\mathbf{X}_i$. If the labels are continuous, this problem is a multi-task regression problem and otherwise a multi-task classification problem. The linear function for the ith task is defined as $\mu_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x}$, where an offset is assumed to be absorbed into $\mathbf{w}_i$. $\mathbf{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_m] \in \mathbb{R}^{d\times m}$ is defined as the parameter matrix.

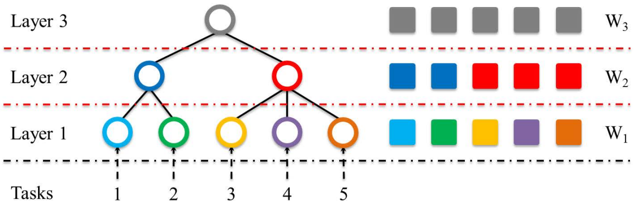

Figure 1: An example of the task tree with 3 layers.

First, we assume that the parameter matrix $\mathbf{W}$ can be decomposed into H component matrices as

$$\mathbf{W} = \sum_{h=1}^{H} \mathbf{W}_h, \qquad (1)$$

where $\mathbf{W}_h = [\mathbf{w}_{h,1}, \cdots, \mathbf{w}_{h,m}] \in \mathbb{R}^{d\times m}$ is the component matrix corresponding to the hth layer and $\mathbf{w}_{h,i}$ is the parameter for the ith task at the hth layer. Then based on the decomposable structure of $\mathbf{W}$, we can define a task tree used to specify the tree structure in MTL as follows.

Definition 1 (Task Tree). *By assuming H layers in the tree structure ($H \geq 2$), i.e. the height of the tree being H, $\{\mathbf{W}_h\}_{h=1}^H$ form a task tree if they satisfy the following conditions:*

- *The model parameters of different tasks corresponding to the hth layer are stored in the component matrix $\mathbf{W}_h$;*
- *For any pair of tasks, for example, the ith and jth tasks, if the corresponding component parameters in the hth layer satisfies that $\mathbf{w}_{h,i} = \mathbf{w}_{h,j}$, then $\mathbf{w}_{h',i} = \mathbf{w}_{h',j}$ should hold in all the above layers where $h' \geq h$.*

Note that the task tree in definition 1 can be any type of tree. Fig. 1 shows an example of the task tree with 3 layers. The leaf nodes in the tree represent tasks, whose relations with each other are represented by a task tree. Each internal node reveals the relations among tasks corresponding to the leaf nodes of a subtree rooted at it. By assuming that the feature dimensionality d equals 1, then each $\mathbf{W}_h$ reduces to a vector for $h \in \{1, 2, 3\}$. The component parameters $\mathbf{W}_h$'s corresponding to each layer are shown at the right side of Fig. 1, where the component parameters with identical values are plotted in the same color.

In order to make the component parameters $\{\mathbf{W}_h\}_{h=1}^H$ form a task tree, we devise sequential constraints on them as

$$|\mathbf{w}_{h-1,i} - \mathbf{w}_{h-1,j}| \succeq |\mathbf{w}_{h,i} - \mathbf{w}_{h,j}|, \forall h \geq 2, \forall i < j, \qquad (2)$$

where $|\cdot|$ denotes the elementwise absolute value and $\succeq$ is the elementwise 'no smaller than' operation. The sequential constraints in Eq. (2) impose a non-increasing order for the pair distance between tasks from the bottom layer to the top one. It is easy to see that once $\mathbf{w}_{h,i} = \mathbf{w}_{h,j}$ for one pair of tasks at the hth layer for some h, then the sequential constraints in Eq. (2) can guarantee that $\mathbf{w}_{h',i} = \mathbf{w}_{h',j}$ for any $h' > h$. So the component parameters $\{\mathbf{W}_h\}_{h=1}^H$ satisfying Eq. (2) can form a task tree.

Moreover, in order to make the component parameters of each pair of tasks possible to be fused, e.g., $\mathbf{w}_{h,i}$ becoming identical to $\mathbf{w}_{h,j}$, we use a structurally sparse regularizer $\Omega(\mathbf{W})$ on the component matrices as

$$\Omega(\mathbf{W}) = \sum_{h=1}^{H} \lambda_h \sum_{i<j}^{d} \|\mathbf{w}_{h,i} - \mathbf{w}_{h,j}\|_2,$$

where $\lambda_1, \cdots, \lambda_H$ are positive regularization parameters to specify the importance of different layers. Based on the use of the ℓ_2 norm, $\Omega(\mathbf{W})$ encourages each pair of the columns $\mathbf{w}_{h,i}$ and $\mathbf{w}_{h,j}$ to be identical. If this happens, the condition in Definition 1 can be satisfied.

By combining the above considerations, we formulate the objective function of the TAT model as

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) + \sum_{h=1}^{H} \lambda_h \sum_{i<j}^{d} \|\mathbf{w}_{h,i} - \mathbf{w}_{h,j}\|_2$$
$$\text{s.t. } |\mathbf{w}_{h-1,i} - \mathbf{w}_{h-1,j}| \succeq |\mathbf{w}_{h,i} - \mathbf{w}_{h,j}|, \forall h \geq 2, \forall i < j, \quad (3)$$

where $\mathcal{L}(\cdot)$ denotes a loss function. In this paper, we consider the square loss $\mathcal{L}(\mathbf{W}) = \sum_{i=1}^{m} \frac{1}{mn_i} \|\mathbf{y}_i - \mathbf{X}_i \sum_{h=1}^{H} \mathbf{w}_{h,i}\|_2^2$, and other loss functions can be accordingly adopted in a similar way.

The regularization parameter λ_h control the strength of the task similarity at the hth layer. A larger λ_h will lead to more identical task parameters in the corresponding component matrix. Therefore, it is intuitive to define a non-decreasing order for the λ_h's from the bottom layer to the top one to help construct the tree structure. In practice, we set $\lambda_h = \phi\lambda_{h-1}$ for $h \geq 2$ with some constant $\phi > 1$.

It is easy to show that problem (3) is non-convex due to the non-convexity of the sequential constraints and the regularizer $\Omega(\mathbf{W})$ is non-smooth, making solving it very challenging. In the next section, we show how to solve problem (3) efficiently.

3. OPTIMIZATION PROCEDURE

In this section, we discuss how to solve problem (3). The main idea is to use the GIST method [8], which involves solving a proximal problem via the ADMM, to solve problem (3).

Since problem (3) is non-convex, we adopt the GIST method to solve it. The GIST method solves a problem with the following form:

$$\min_{\mathbf{W}} f(\mathbf{W}) + r(\mathbf{W}),$$

where $f(\cdot)$ is convex and Lipschitz continuous, and $r(\cdot)$ can be non-convex. In order to apply the GIST method, we define $f(\mathbf{W}) = \mathcal{L}(\mathbf{W})$ and $r(\mathbf{W})$ as a non-convex extended real-value function with the formulation as

$$r(\mathbf{W}) = \begin{cases} \Omega(\mathbf{W}), & \text{if } |\mathbf{w}_{h-1,i} - \mathbf{w}_{h-1,j}| \succeq |\mathbf{w}_{h,i} - \mathbf{w}_{h,j}| \\ & \text{for any } h \geq 2,\ i < j, \\ +\infty, & \text{otherwise.} \end{cases}$$

According to [8], the proximal operator at the $(k+1)$th iteration in the GIST method is

$$\mathbf{W}^{(k+1)} = \arg\min_{\mathbf{W}} f\left(\mathbf{W}^{(k)}\right) + \frac{\tau_k}{2} \sum_h \|\mathbf{W}_h - \mathbf{W}_h^{(k)}\|_F^2 + r(\mathbf{W})$$
$$+ \sum_h \left\langle \nabla_{\mathbf{W}_h} f(\mathbf{W}^{(k)}), \mathbf{W}_h - \mathbf{W}_h^{(k)} \right\rangle, \quad (4)$$

where $\mathbf{W}^{(k)}$ denotes the estimation in the kth iteration with $\mathbf{W}_h^{(k)}$ as its hth component matrix, $\nabla_{\mathbf{W}_h} f(\mathbf{W}^{(k)})$ denotes the gradient of $f(\mathbf{W})$ with respect to $\mathbf{W}_h$ at $\mathbf{W}^{(k)}$, and τ_k is determined via a line search method as described in Algorithm 1. By omitting the constant part, problem (4) can be simplified as

$$\mathbf{W}^{(k+1)} = \arg\min_{\mathbf{W}} \frac{\tau_k}{2} \sum_h \|\mathbf{W}_h - \mathbf{V}_h\|_F^2 + r(\mathbf{W}), \quad (5)$$

where $\mathbf{V}_h = \mathbf{W}_h^{(k)} - \frac{1}{\tau_k} \nabla_{\mathbf{W}_h} f(\mathbf{W}^{(k)})$. For all $h \in \mathbb{N}_H$, the gradients $\nabla_{\mathbf{W}_h} f(\mathbf{W}^{(k)})$'s are identical, where the ith column of $\nabla_{\mathbf{W}_h} f(\mathbf{W}^{(k)})$ can be easily computed as $\frac{2}{mn_i} \mathbf{X}_i^T (\mathbf{X}_i \mathbf{w}_i^{(k)} - \mathbf{y}_i)$ for any h. The GIST algorithm is presented in Algorithm 1.

Algorithm 1 The GIST algorithm for solving problem (3).

```
Input: $\mathbf{X}$, $\mathbf{Y}$, $H$;
Output: $\mathbf{W} = \sum_h \mathbf{W}_h$;
Initialize $\mathbf{W}_1^{(0)}, \cdots, \mathbf{W}_H^{(0)}, \eta > 1, \tau_{min}, \tau_{max}, \varphi \in (0,1), k = 0$;
repeat
    $\tau_k \in [\tau_{min}, \tau_{max}]$;
    repeat
        Solve the proximal problem (5) with $\tau_k$;
        $\tau_k = \eta\tau_k$;
    until $F(\mathbf{W}^{(k+1)}) \leq F(\mathbf{W}^{(k)}) - \frac{\varphi}{2}\tau_k \|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\|_2^2$;
    $k := k + 1$;
until Some convergence criterion is satisfied;
```

3.1 Solving Proximal Problem (5)

In the GIST algorithm, we only need to solve the proximal problem (5). Since $r(\cdot)$ is an extended real-value function, problem (5) can be reformulated as

$$\min_{\mathbf{W}} \frac{\tau}{2} \sum_h \|\mathbf{W}_h - \mathbf{V}_h\|_F^2 + \sum_h \lambda_h \sum_{i<j} \|\mathbf{w}_{h,i} - \mathbf{w}_{h,j}\|_2,$$
$$\text{s.t. } |\mathbf{w}_{h-1,i} - \mathbf{w}_{h-1,j}| \succeq |\mathbf{w}_{h,i} - \mathbf{w}_{h,j}|,\ h \geq 2,\ i < j, \quad (6)$$

where for notational simplicity we omit the index k. Unfortunately, problem (6) is still non-convex due to the sequential constraints and it is still non-smooth. By introducing new variables, we use the ADMM method to solve problem (6).

We define an auxiliary sparse matrix

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots \\ 1 & 0 & -1 & 0 & \cdots \\ \vdots & & \ddots & & \vdots \end{bmatrix} \in \mathbb{R}^{\frac{m(m-1)}{2} \times m},$$

where each row of $\mathbf{C}$ contains only two non-zero entries 1 and -1 at corresponding locations. By defining new variables $\{\mathbf{Q}_h\}_{h=1}^H$ as $\mathbf{Q}_h = \mathbf{C}\mathbf{W}_h^T$, we can get

$$\|\mathbf{Q}_h\|_{1,2} = \sum_{i<j} \|\mathbf{w}_{h,i} - \mathbf{w}_{h,j}\|_2.$$

Similarly, by introducing variables $\{\mathbf{P}_h\}_{h=1}^H$ as a copy of $\{\mathbf{Q}_h\}_{h=1}^H$, problem (6) can be reformulated as

$$\min_{\mathbf{W}} \frac{\tau}{2} \sum_h \|\mathbf{W}_h - \mathbf{V}_h\|_F^2 + \sum_h \lambda_h \|\mathbf{P}_h\|_{1,2}$$
$$\text{s.t. } \mathbf{P}_h = \mathbf{Q}_h,\ \mathbf{Q}_h = \mathbf{C}\mathbf{W}_h^T,\ \forall h,$$
$$|\mathbf{q}_{h-1}^i| \succeq |\mathbf{q}_h^i|, \forall h \geq 2,\ i \in \mathbb{N}_{m(m-1)/2}. \quad (7)$$

In order to use the ADMM method, we define the augmented Lagrangian function as

$$\bar{\mathcal{L}}(\mathbf{W}, \mathbf{P}, \mathbf{Q}) = \frac{\tau}{2} \sum_h \|\mathbf{W}_h - \mathbf{V}_h\|_F^2 + \frac{\rho}{2} \|\mathbf{Q}_h - \mathbf{C}\mathbf{W}_h^T\|_F^2$$
$$+ \sum_h \text{tr}\left(\mathbf{\Theta}_h^T(\mathbf{P}_h - \mathbf{Q}_h)\right) + \frac{\rho}{2} \sum_h \|\mathbf{P}_h - \mathbf{Q}_h\|_F^2$$
$$+ \sum_h \text{tr}\left(\mathbf{\Gamma}_h^T(\mathbf{Q}_h - \mathbf{C}\mathbf{W}_h^T)\right) + \sum_h \lambda_h \|\mathbf{P}_h\|_{1,2},$$

where $\mathbf{P}$ denotes the set of $\{\mathbf{P}_h\}_{h=1}^H$, $\mathbf{Q}$ denotes the set of $\{\mathbf{Q}_h\}_{h=1}^H$, $\{\mathbf{\Theta}_h\}_{h=1}^H$ and $\{\mathbf{\Gamma}_h\}_{h=1}^H$ acts as Lagrangian multipliers, and ρ is a penalty parameter. Then we need to solve the following problem as

$$\min_{\mathbf{W},\mathbf{P},\mathbf{Q}} \bar{\mathcal{L}}(\mathbf{W}, \mathbf{P}, \mathbf{Q}), \text{ s.t. } |\mathbf{q}_{h-1}^i| \succeq |\mathbf{q}_h^i|, h \geq 2, i \in \mathbb{N}_{m(m-1)/2}. \quad (8)$$

In the ADMM algorithm whose procedure is presented in Algorithm 2, problem (8) can be solved alternatively with respect to $\mathbf{W}$, $\mathbf{P}$, and $\mathbf{Q}$. In the following, we discuss how to solve those three subproblems corresponding to steps 4-6 in the ADMM algorithm.

Algorithm 2 The ADMM algorithm for solving problem (8).

```
Input: $\mathbf{X}$, $\mathbf{Y}$, $H$, $\mathbf{W}_h^{(0)}$;
Output: $\mathbf{W} = \sum_h \mathbf{W}_h$;
Initialize $\{\mathbf{P}_h^{(0)}, \mathbf{Q}_h^{(0)}, \mathbf{\Theta}_h^{(0)}, \mathbf{\Gamma}_h^{(0)}\}_{h=1}^H$;
Set $\rho = 0.1$ and $t = 0$;
repeat
    Solve $\mathbf{W}_h^{(t+1)}$ with fixed $\mathbf{P}_h^{(t)}$ and $\mathbf{Q}_h^{(t)}$;
    Solve $\mathbf{P}_h^{(t+1)}$ with fixed $\mathbf{W}_h^{(t)}$ and $\mathbf{Q}_h^{(t)}$;
    Solve $\mathbf{Q}_h^{(t+1)}$ with fixed $\mathbf{W}_h^{(t)}$ and $\mathbf{P}_h^{(t)}$;
    Update $\mathbf{\Theta}_h^{(t+1)} = \mathbf{\Theta}_h^{(t)} + \rho\left(\mathbf{P}_h^{(t)} - \mathbf{Q}_h^{(t)}\right)$;
    Update $\mathbf{\Gamma}_h^{(t+1)} = \mathbf{\Gamma}_h^{(t)} + \rho\left(\mathbf{Q}_h^{(t)} - \mathbf{C}\left(\mathbf{W}_h^{(t)}\right)^T\right)$;
    $t := t + 1$;
until Some convergence criterion is satisfied;
```

3.1.1 Solving Problem (8) w.r.t. $\mathbf{W}$

With fixed $\mathbf{P}$ and $\mathbf{Q}$, we need to solve the following subproblem with respect to $\mathbf{W}$ as

$$\min_{\mathbf{W}} \frac{\tau}{2}\sum_h \|\mathbf{W}_h - \mathbf{V}_h\|_F^2 + \frac{\rho}{2}\sum_h \|\mathbf{Q}_h - \mathbf{C}\mathbf{W}_h^T\|_F^2 + \sum_h \operatorname{tr}\left(\mathbf{\Gamma}_h^T\left(\mathbf{Q}_h - \mathbf{C}\mathbf{W}_h^T\right)\right). \tag{9}$$

Problem (9) can be decomposed into H separable problems, each of which can be solved analytically based on the stationary condition where the solution can computed as

$$\mathbf{W}_h = \left(\tau\mathbf{V}_h + \left(\rho\mathbf{Q}_h + \mathbf{\Gamma}_h\right)^T\mathbf{C}\right)\left(\tau\mathbf{I} + \rho\mathbf{C}^T\mathbf{C}\right)^{-1}. \tag{10}$$

Since $(\tau\mathbf{I} + \rho\mathbf{C}^T\mathbf{C})^{-1}$ is a constant matrix, it can be pre-computed and stored, leading to efficient computation of Eq. (10).

3.1.2 Solving Problem (8) w.r.t. $\mathbf{P}$

With fixed $\mathbf{W}$ and $\mathbf{Q}$, $\mathbf{P}$ can be obtained by solving the following problem:

$$\min_{\mathbf{P}} \frac{\rho}{2}\sum_h \|\mathbf{P}_h - \mathbf{S}_h\|_F^2 + \sum_h \lambda_h\|\mathbf{P}_h\|_{1,2}, \tag{11}$$

where $\mathbf{S}_h = \mathbf{Q}_h - \frac{1}{\rho}\mathbf{\Theta}_h$. Problem (11) can be decomposed into $\frac{1}{2}Hm(m-1)$ independent problems with one problem corresponding to a row of matrix $\mathbf{P}_h$ formulated as

$$\min_{\mathbf{p}_h^i} \frac{\rho}{2}\|\mathbf{p}_h^i - \mathbf{s}_h^i\|_2^2 + \lambda_h\|\mathbf{p}_h^i\|_2, \tag{12}$$

where $\mathbf{s}_h^i$ is the ith row of $\mathbf{S}_h$. Problem (12) is widely studied in the group Lasso and admits a closed-form solution as

$$\mathbf{p}_h^i = \left(1 - \frac{\lambda_h}{\rho\|\mathbf{s}_h^i\|_2}\right)_+ \mathbf{s}_h^i, \tag{13}$$

where $(x)_+ = \max(0, x)$ is a thresholding operator.

3.1.3 Solving Problem (8) w.r.t. $\mathbf{Q}$

The remaining problem is to solve $\mathbf{Q}$, which is a key step. In problem (8), the sequential constraints bring challenges to solve it. With the given $\mathbf{W}$ and $\mathbf{P}$, we can update $\mathbf{Q}$ by solving the following problem as

$$\min_{\mathbf{Q}} \frac{\rho}{2}\sum_h \|\mathbf{Q}_h - \mathbf{U}_h\|_F^2, \text{ s.t. } |\mathbf{q}_{h-1}^i| \succeq |\mathbf{q}_h^i|, \forall h \geq 2, \tag{14}$$

where $\mathbf{U}_h = \frac{1}{2}\left(\mathbf{P}_h + \mathbf{C}\mathbf{W}_h^T - \frac{1}{\rho}\mathbf{\Gamma}_h + \frac{1}{\rho}\mathbf{\Theta}_h\right)$. Obviously problem (14) can be decomposed into $\frac{1}{2}m(m-1)d$ independent problems each of which is formulated as

$$\min_{\{q_{h,j}^i\}_{h=1}^H} \sum_h \left(q_{h,j}^i - \hat{u}_{h,j}^i\right)^2, \text{ s.t. } |q_{h-1,j}^i| \geq |q_{h,j}^i|, \forall h \geq 2, \tag{15}$$

where $q_{h,j}^i$ and $\hat{u}_{h,j}^i$ are the (i,j)th elements of $\mathbf{Q}_h$ and $\mathbf{U}_h$ respectively. For simplicity, we omit the scripts i and j in problem (15), and get

$$\min_{\{q_h\}_{h=1}^H} \sum_h \left(q_h - \hat{u}_h\right)^2, \text{ s.t. } |q_{h-1}| \geq |q_h|, \forall h \geq 2. \tag{16}$$

Note that we have

$$(q_h - \hat{u}_h)^2 = (\text{sign}(q_h)|q_h| - \hat{u}_h)^2 = (|q_h| - \text{sign}(q_h)\hat{u}_h)^2,$$

where $\text{sign}(\cdot)$ denotes the sign function. q_h must have the same sign as $\hat{u}_h$ since otherwise we can change the sign of q_h to achieve a lower objective function value for problem (16). So by defining $\bar{q}_h = |q_h|$ and $\bar{u}_h = |\hat{u}_h|$, problem (16) is equivalent to the following problem:

$$\min_{\{\bar{q}_h\}_{h=1}^H} \sum_h \left(\bar{q}_h - \bar{u}_h\right)^2, \text{ s.t. } \bar{q}_1 \geq \bar{q}_2 \geq \cdots \geq \bar{q}_H. \tag{17}$$

After solving problem (17), we can recover the solution of problem (16) as $q_h = \text{sign}(\hat{u}_h)\bar{q}_h$. Note that problem (17) is convex. In the next section, we show that problem (17) can be solved efficiently in $O(H)$ complexity.

3.2 Solving Problem (17)

In each iteration of the ADMM method, problem (17) needs to be solved for $m(m-1)d/2$ times, hence it requires a very efficient solution.

In the following analysis, a sequence $(\bar{q}_1', \cdots, \bar{q}_H')$ is said to be better (worse) than another sequence $(\bar{q}_1'', \cdots, \bar{q}_H'')$ for problem (17) if both the sequences are feasible for problem (17), i.e. satisfying the sequential constraints, and the objective value of problem (17) at $(\bar{q}_1', \cdots, \bar{q}_H')$ is smaller (larger) than that at $(\bar{q}_1'', \cdots, \bar{q}_H'')$.

In order to solve problem (17), we first introduce some useful lemmas to reveal interesting properties.

LEMMA 1. *Problem (17) has the following properties:*

1. *if $\bar{u}_1 \geq \bar{u}_2 \geq \cdots \geq \bar{u}_H$, then the optimal solution $(\bar{q}_1^*, \bar{q}_2^*, \cdots, \bar{q}_H^*)$ is $(\bar{u}_1, \bar{u}_2, \cdots, \bar{u}_H)$;*
2. *if $\bar{u}_1 \leq \bar{u}_2 \leq \cdots \leq \bar{u}_H$, then the optimal solution $(\bar{q}_1^*, \bar{q}_2^*, \cdots, \bar{q}_H^*)$ is $(u^*, \cdots, u^*)|_H$, where $u^* = \frac{1}{H}\sum_{h=1}^H \bar{u}_h$.*

LEMMA 2. *For any sequence $(\bar{u}_1, \cdots, \bar{u}_H)$, if the optimal solution of problem (17) is $(u^*, \cdots, u^*)|_H$, then for any $u^* \geq u' \geq b_1$ or $b_H \geq u' \geq u^*$, the sequence $(b_1, \cdots, b_H)$, where $b_1 \geq \cdots \geq b_H$, is not better than $(u', \cdots, u')|_H$.*

Based on the properties revealed in Lemmas 1 and 2, we can present the following important theorem.

THEOREM 1. *For any two sequences $(\bar{u}_1, \cdots, \bar{u}_l)$ and $(\bar{u}_{l+1}, \cdots, \bar{u}_n)$ which define two instances of problem (17), if the optimal solutions for them are $(\dot{u}^*, \cdots, \dot{u}^*)|_l$ and $(\ddot{u}^*, \cdots, \ddot{u}^*)|_{n-l}$, respectively, then*

1. *if $\dot{u}^* \geq \ddot{u}^*$, the optimal solution for the concatenated sequence $(\bar{u}_1, \cdots, \bar{u}_l) \bowtie (\bar{u}_{l+1}, \cdots, \bar{u}_n)$, i.e. $(\bar{u}_1, \cdots, \bar{u}_l, \bar{u}_{l+1}, \cdots, \bar{u}_n)$, is $(\dot{u}^*, \cdots, \dot{u}^*)|_l \bowtie (\ddot{u}^*, \cdots, \ddot{u}^*)|_{n-l}$;*

2. *otherwise, the optimal solution for the concatenated sequence is $(u^*, \cdots, u^*)|_n$, where $u^* = \frac{1}{n}\sum_{i=1}^n \bar{u}_i$.*

Theorem 1 implies that we can obtain the solution of any sequence $(\bar{u}_1, \cdots, \bar{u}_n)$ in problem (17), if the sequence is a concatenation from two sub-sequences, where the optimal solutions corresponding to the two sub-sequences take the form that the entries in a solution have the same value. Finally, based on Theorem 1, we devise Algorithm 3 to solve problem (17) with its correctness guaranteed by the following theorem.

THEOREM 2. *For any input sequence $(\bar{u}_1, \cdots, \bar{u}_H)$ in problem (17), Algorithm 3 can find the optimal solution.*

Algorithm 3 The algorithm for solving problem (17).

```
Input: $(\bar{u}_1, \cdots, \bar{u}_H)$;
Output: $(q_1^*, \cdots, q_H^*)$;
 1: Scan the sequence $(\bar{u}_1, \cdots, \bar{u}_H)$ once to split it into $K$ non-decreasing
    sub-sequences $(S_1, \cdots, S_K)$ and calculate the solutions for those sub-
    sequences based on Lemma 1;
 2: Push $S_1$ into a stack;
 3: for $k = 2 : K$ do
 4:     Push $S_k$ into the stack;
 5:     while there are at least two sequences in the stack do
 6:         Pop the first and second sequences from the stack and denote the
            solutions for them as $\ddot{u}^*$ and $\dot{u}^*$ separately;
 7:         if $\dot{u}^* < \ddot{u}^*$ then
 8:             Concatenate the two sequences under the second condition in
                Theorem 1 and then push the concatenated sequence into the
                stack;
 9:         else
10:             Push the two sequences into the stack without any operation;
11:             Break;
12:         end if
13:     end while
14: end for
15: Concatenate the solutions of the sequences in the stack from bottom to
    top and output the concatenated solution;
```

3.3 Time Complexity

In this section, we analyze the time complexity of the whole optimization procedure for solving the TAT model.

We first discuss the time complexity of Algorithm 3, since it is the inner most one. In Algorithm 3, step 1 only needs to scan the input sequence once and thus it needs $O(H)$ time. Although there exist two loops from step 3 to 14, the maximum number of the concatenation operations in step 8 is $K-1$ and this step costs $O(H)$. For the concatenation operation, the computation of the analytical solution in Theorem 1 is very efficient, since it only needs to compute the average of the entries in two sequences. We can record the average and the number of the entries in each sequence, and then each concatenation operation only needs $O(1)$ time. Therefore, Algorithm 3 can be executed in $O(H)$ time complexity.

In Algorithm 2, the computation of the solution for $\mathbf{W}$ with each component $\mathbf{W}_h$ in Eq. (10) takes $O\left(m^3Hd\right)$ time. The computation of the solution for matrix $\mathbf{P}$ needs $O(m^2H)$ time. For solving $\mathbf{Q}$, Algorithm 3 needs to be executed for $m(m-1)d/2$ times, hence the time cost is $O(m^2Hd)$. So the time complexity of each iteration in Algorithm 2 is $O\left(m^3Hd\right)$. By assuming that Algorithms 1 and 2 need N_1 and N_2 iterations to converge, respectively, the total time complexity for solving the TAT model is $O\left(N_1N_2m^3Hd\right)$.

In our experiments, we empirically find that both Algorithm 1 and Algorithm 2 need very small numbers of iterations to converge. Therefore, the overall algorithm for solving the TAT model is very efficient. Moreover, according to Section 3.1, the subproblems for $\mathbf{W}$, $\mathbf{P}$, and $\mathbf{Q}$ can be decomposed into a large number of independent problems, which can be parallelized to further improve the efficiency.

4. THEORETICAL ANALYSIS

In this section, we provide theoretical analysis for the TAT model. For notational simplicity, we assume that the numbers of training samples for all the tasks are the same and denote it by n. The general case that different tasks have different numbers of training samples can be similarly analyzed.

We assume that the true relation between the data sample and its label is a linear function plus a Gaussian noise, which is defined as $y_{ji} = (\mathbf{x}_j^{(i)})^T\mathbf{w}_i^* + \epsilon_{ji}, i \in \mathbb{N}_m, j \in \mathbb{N}_n$, where $\mathbf{x}_j^{(i)}$ is the jth data point of the ith task with y_{ji} as its label, $\mathbf{W}^* = [\mathbf{w}_1^*, \ldots, \mathbf{w}_m^*]$ is the true parameter matrix, and all the ϵ_{ji}'s are independent Gaussian noises sampled from $\mathcal{N}(0, \sigma^2)$. We assume that $\mathbf{W}^*$ can be decomposed into H true component matrices $\mathbf{W}_1^*, \ldots, \mathbf{W}_H^*$ as $\mathbf{W}^* = \sum_{h=1}^H \mathbf{W}_h^*$, where those component matrices satisfy the sequential constraints in problem (3). We define $\mathbf{f}_i^* = \mathbf{X}_i\mathbf{w}_i^*$ and $\mathbf{y}_i = \mathbf{f}_i^* + \boldsymbol{\epsilon}_i$ for $i \in \mathbb{N}_m$, where $\boldsymbol{\epsilon}_i = [\epsilon_{1i}, \ldots, \epsilon_{ni}]^T$. Let $\mathbf{X} \in \mathbb{R}^{dm\times mn}$ be a diagonal block matrix with $\mathbf{X}_i^T$ as the ith block for $i \in \mathbb{N}_m$. We define a vectorization operator $\mathrm{vec}(\cdot)$ over an arbitrary matrix $\mathbf{X} \in \mathbb{R}^{d\times m}$ to concatenate its columns as $\mathrm{vec}(\mathbf{X}) = [\mathbf{x}_1^T, \cdots, \mathbf{x}_m^T]^T$. Let $\mathbf{F}^* = [\mathbf{f}_1^*, \ldots, \mathbf{f}_m^*] \in \mathbb{R}^{n\times m}$.

For any matrix $\mathbf{X} \in \mathbb{R}^{d\times m}$, we define $E(\mathbf{X}) = \{(i,j)|\mathbf{x}_i \neq \mathbf{x}_j, i, j \in \mathbb{N}_m\}$ and its complement $E_c(\mathbf{X}) = \{(i,j)|\mathbf{x}_i = \mathbf{x}_j, i \in \mathbb{N}_m, j \in \mathbb{N}_m, i \neq j\}$. For any matrix $\mathbf{X} \in \mathbb{R}^{d\times m}$, since each pair (i,j) corresponds to one row in $\mathbf{CX}^T \in \mathbb{R}^{\frac{m(m-1)}{2}\times d}$, which is $\mathbf{x}_i^T - \mathbf{x}_j^T$, the projections of the rows in $\mathbf{CX}^T$ onto the set $E(\mathbf{X})$, denoted by $\left(\mathbf{CX}^T\right)^{E(\mathbf{X})}$, consist of the rows with non-zero ℓ_2 norms in $\mathbf{CX}^T$, and similar definitions can be made for the set $E_c(\mathbf{X})$. We define $D(\mathbf{X})$ as the set of indices for distinct column vectors in $\mathbf{X}$, i.e., for any $i, j \in D(\mathbf{X})$, $\mathbf{x}_i \neq \mathbf{x}_j$. We denote by $\mathbf{X}^{D(\mathbf{X})}$ the projection of the columns of $\mathbf{X}$ onto the set $D(\mathbf{X})$ and the complement of $D(\mathbf{X})$ by $D_c(\mathbf{X})$.

In order to analyze the TAT model, we need the following assumption.

ASSUMPTION 1. *Let $\hat{\mathbf{W}} = \sum_{h=1}^H \hat{\mathbf{W}}_h$ be the optimal solution of problem (3). For any matrix $\mathbf{W} = \sum_{h=1}^H \mathbf{W}_h \in \mathbb{R}^{d\times m}$ and $h \in \mathbb{N}_H$, we define matrix $\boldsymbol{\Delta}_h$ as $\boldsymbol{\Delta}_h = \mathbf{W}_h - \hat{\mathbf{W}}_h$ and matrix $\boldsymbol{\Gamma}_h$ as $\boldsymbol{\Gamma}_h = \mathbf{CW}_h^T - \mathbf{C}\hat{\mathbf{W}}_h^T$. Let $\boldsymbol{\Delta} = \sum_{h=1}^H \boldsymbol{\Delta}_h$. We assume that there exist positive scalars β_h, $\theta_h \geq 1$, and $\gamma_h \geq 1$ such that*

$$\beta_h = \min_{\boldsymbol{\Delta}_h \neq \mathbf{0}} \frac{\|\mathbf{X}^T\mathrm{vec}(\boldsymbol{\Delta})\|_2}{\sqrt{mn}\|\boldsymbol{\Delta}_h^{D(\mathbf{W}_h)}\|_F},$$

$$\|\boldsymbol{\Delta}_h\|_F = \theta_h\|\boldsymbol{\Delta}_h^{D(\mathbf{W}_h)}\|_F,$$

$$\|\boldsymbol{\Gamma}_h\|_{1,2} = \gamma_h\|\boldsymbol{\Gamma}_h^{E(\mathbf{W}_h)}\|_{1,2}.$$

Assumption 1 refers to the restricted eigenvalue assumption as introduced in [20] and similar assumptions are commonly used in the MTL literature, e.g., [4, 12]. Based on Assumption 1, we can analyze the TAT model in the following theorem.[1]

THEOREM 3. *Let $\hat{\mathbf{W}} = \sum_{h=1}^H \hat{\mathbf{W}}_h$ be the optimal solution of problem (3) and define $\mathcal{C} = \sum_{h'=1}^H \frac{\lambda_{h'}(\theta_{h'}+1)}{\beta_{h'}}$. If the regulariza-*

[1] Due to page limitations, we put the proof at http://www.comp.hkbu.edu.hk/~leihan/.

tion parameters λ_h for any $h \in \mathbb{N}_H$ satisfies[2]

$$\lambda_h \geq \frac{2\sigma\sqrt{m+\delta/d}}{m(m-1)n}, \tag{18}$$

then under Assumption 1, the following results hold with probability at least $1 - \exp(-\frac{1}{2}(\delta - dm\log(1+\frac{\delta}{dm})))$:

$$\|\mathbf{X}^T\mathrm{vec}(\hat{\mathbf{W}}) - \mathrm{vec}(\mathbf{F}^*)\|_2^2 \leq m(m-1)^2nd\mathcal{C}^2, \tag{19}$$

$$\|\hat{\mathbf{W}}_h - \mathbf{W}_h^*\|_F \leq \frac{\theta_h(m-1)\sqrt{d}\mathcal{C}}{\beta_h}, \tag{20}$$

$$\|\mathbf{C}\hat{\mathbf{W}}_h^T - \mathbf{C}(\mathbf{W}_h^*)^T\|_{1,2} \leq \frac{\gamma_h(m-1)^2d\mathcal{C}}{\beta_h}. \tag{21}$$

In addition, if the following condition holds for $h \in \mathbb{N}_H$:

$$\min_{(i,j)\in E(\mathbf{W}_h^*)} \left\| \left[\mathbf{C}(\mathbf{W}_h^*)^T\right]^{(i,j)} \right\|_2 > \frac{2d\gamma_h(m-1)^2\mathcal{C}}{\beta_h}, \tag{22}$$

where $\left[\mathbf{C}(\mathbf{W}_h^*)^T\right]^{(i,j)}$ denotes one row in $\mathbf{C}(\mathbf{W}_h^*)^T$ corresponding to the pair (i,j), then with probability at least $1 - \exp(-\frac{1}{2}(\delta - dm\log(1+\frac{\delta}{dm})))$, the following set

$$\hat{E}_h = \left\{(i,j) \middle| \left\| \left(\mathbf{C}\hat{\mathbf{W}}_h\right)^{(i,j)} \right\|_2 > \frac{d\gamma_h(m-1)^2\mathcal{C}}{\beta_h} \right\} \tag{23}$$

can recover the true task structure $E(\mathbf{W}_h^*)$ at the hth layer of the task tree, i.e. $\hat{E}_h = E(\mathbf{W}_h^*)$ and $(\hat{E}_h)_c = E_c(\mathbf{W}_h^*)$.

Theorem 3 provides important theoretical guarantees for the TAT model. Specifically, those bounds measure how well our model can approximate the ground truth of the component matrix $\mathbf{W}_h^*$ as well as the true parameter matrix $\mathbf{W}^* = \sum_{h=1}^H \mathbf{W}_h^*$. Moreover, if the assumptions in Eq. (22) can be satisfied, the TAT model can recover the true task tree with high probability based on Eq. (23).

5. RELATED WORK

The proposed TAT model is related to some hierarchical MTL methods [15, 9, 28, 12], since those works assume the tasks are organized as hierarchies, which in some sense are a bit similar to the layers in the proposed task tree. However, the hierarchical structure proposed in those works cannot be organized as a tree and different hierarchies in them are independent of each other, which is totally different from the proposed TAT model where different layers in the task tree follow the sequential constraints in Eq. (2).

The TAT model is also related to some task grouping methods [14, 16, 18]. The task grouping methods aim to learn clusters for tasks, and therefore they are just corresponding to the bottom layer of the task tree in the TAT model.

6. EXPERIMENTS

In this section, we conduct empirical experiments on both synthetic and real-world problems to study the proposed TAT method. We compare the TAT method with several state-of-the-art MTL models, including the multi-task feature learning (MTFL) model [19],[3] the dirty model (Dirty) [15],[3] the Cascade model [28], the clustered multi-task learning (CMTL) model [14],[4] the grouping and overlap MTL (GOMTL) model [18], and the multi-level task grouping (MeTaG) model [12]. Among these competitors, the MTFL method learns common feature representation for multiple tasks, the Dirty and Cascade models are representatives of the hierarchical MTL models, the CMTL and GOMTL models are belonging to the task grouping approach, and the MeTaG model is a combination of those two types.

The regularization parameters of all the methods in comparison are determined via the validation dataset with the set of candidate values as $\{10^{-8}, 10^{-7}, \cdots, 10^3\}$. Moreover, in the same way we choose ϕ, which defines $\frac{\lambda_h}{\lambda_{h-1}}$, from a set $\{1.2, 2, 10\}$ for the TAT model. In addition to the regularization parameters, some of the competitors include some additional hyper-parameters, including the number of groups in the CMTL method, the dimension of latent subspace in the GOMTL method, the number of cascades in the Cascade method, the number of levels in the MeTaG method, and the number of layers in our TAT model. Those hyper-parameters are selected from a candidate set $\{1, 2, \cdots, 10\}$ because empirically we find that any value larger than 10 will lead to worse performance for all the models under all the settings. In all the experiments, we use the least square solution to initialize the parameter matrix in the TAT model as a warm start.

6.1 Results on Synthetic Data

We first evaluate all the models on synthetic data. In order to simulate different structures of the task tree, we adopt the binary tree structure and vary the height of the tree, which equals the number of layers, according to the number of tasks. The number of tasks m is equal to $m = 2^{H^*-1}$, where H^* denotes the height of the tree. We vary the number of tasks as $m \in \{4, 8, 16, 32, 64\}$ and the height of the tree $H^* \in \{3, 4, 5, 6, 7\}$ accordingly. Fig. 2 shows two examples of the task tree when $H^* = 4$ and $H^* = 6$. We set the feature dimensionality to be $d = 100$. Then, based on the task tree, we generate each component matrix $\mathbf{W}_h^*$ according to the hth layer in the task tree. If there are k internal nodes in the hth layer, the columns in $\mathbf{W}_h^*$ corresponding to the tasks, whose corresponding leaf nodes belong to the subtree rooted at each internal node, form a group and will have the same value. To achieve this, we generate the component matrices from the top layer to the bottom one. For the component matrix corresponding to the top layer, all the columns are set to be identical and the entries in the columns are generated from $\mathcal{N}(1,1)$. Then, for any layer below, the corresponding component matrix is generated based on the component matrix in the upper layer by adding different noises corresponding to different groups in current layer. The noises in the column are absolute values of samples from $\mathcal{N}(0, 0.2)$. Finally, we generate the parameter matrix $\mathbf{W}^*$ as $\mathbf{W}^* = \sum_h \mathbf{W}_h^*$. Based on $\mathbf{W}^*$, the label vector $\mathbf{y}_i$ for the ith task is generated as $\mathbf{y}_i = \mathbf{X}_i\mathbf{w}_i^* + \boldsymbol{\epsilon}_i$, where each entry in $\mathbf{X}_i$ is generated from $\mathcal{N}(0,1)$ and $\boldsymbol{\epsilon}_i$ is a vector of noises with its entries generated from $\mathcal{N}(0,1)$. We use the mean square error (MSE) to measure the performance of the estimation $\hat{\mathbf{W}}$, which is defined as MSE $= \frac{1}{mn}\sum_{i=1}^m(\hat{\mathbf{w}}_i - \mathbf{w}_i^*)^T\mathbf{X}_i^T\mathbf{X}_i(\hat{\mathbf{w}}_i - \mathbf{w}_i^*)$. In each setting, we generate $n_{tr} = 100$ samples for training, $n_{te} = 100$ samples for testing, and $n_v = 100$ samples as a validation set to select the hyper-parameters in all the methods.

Table 2 shows the average performance of all the methods over 10 random simulations. As shown in Table 2, the MTFL has high estimation errors under all the settings. One possible reason is that it simply learns common feature representations but cannot capture the complex task tree structure. The task grouping models, i.e. the CMTL and GOMTL methods, achieve lower estimation errors than the hierarchial MTL models, i.e. the Dirty and Cascade methods, since the task structure in each level of the task tree can be viewed as task groups. The MeTaG model is a combination of

[2] In the TAT model, we assume an increasing order for $\{\lambda_1, \cdots, \lambda_H\}$ and hence only λ_1 needs to satisfy Eq. (18).

[3] `http://www.yelab.net/software/`

[4] `http://cbio.ensmp.fr/~ljacob/documents/cmtl-code.tgz`

Table 2: The average MSE's of different methods over 10 random simulations on synthetic data (mean ± standard derivative). The highlighted numbers stand for the best results under the significance t-test with 95% confidence.

Data	Num. of samples	$n_{tr}, n_{te}, n_v = 100$	$n_{tr}, n_{te}, n_v = 100$	$n_{tr}, n_{te}, n_v = 100$	$n_{tr}, n_{te}, n_v = 100$	$n_{tr}, n_{te}, n_v = 100$
	Num. of features	$d = 100$	$d = 100$	$d = 100$	$d = 100$	$d = 100$
	Num. of tasks	$m = 4$	$m = 8$	$m = 16$	$m = 32$	$m = 64$
	Num. of layers	$H^* = 3$	$H^* = 4$	$H^* = 5$	$H^* = 6$	$H^* = 7$
Model	MTFL	2.1878±0.1316	2.3961±0.1322	2.8297±0.1429	6.4354±0.1565	10.2075±0.2035
	Dirty	0.9972±0.0611	1.0694±0.0763	1.1847±0.0772	1.3951±0.0754	1.9000±0.0661
	Cascade	0.8950±0.0475	0.9049±0.0435	0.9173±0.0348	0.9487±0.0230	0.9690±0.0395
	CMTL	0.1632±0.0676	0.2689±0.0190	0.2633±0.0093	0.3132±0.0129	0.3684±0.0069
	GOMTL	0.1807±0.0163	0.2236±0.0366	0.4767±0.0337	0.4976±0.0481	0.3444±0.0168
	MeTaG	0.2123±0.0177	0.2696±0.0178	0.3427±0.0225	0.3000±0.0136	0.3446±0.0087
	TAT	**0.1548±0.0113**	**0.2020±0.0165**	**0.2401±0.0105**	**0.2647±0.0101**	**0.3352±0.0066**
	Best H in TAT	$H = \mathbf{3}$	$H = \mathbf{4}$	$H = \mathbf{5}$	$H = \mathbf{6}$	$H = \mathbf{6}$

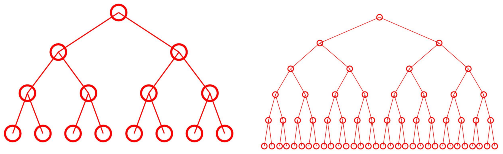

Figure 2: Two examples of the task tree in the synthetic data, where $m = 2^{H^*-1}$. Left: $m = 8, H^* = 4$; right: $m = 32, H^* = 6$.

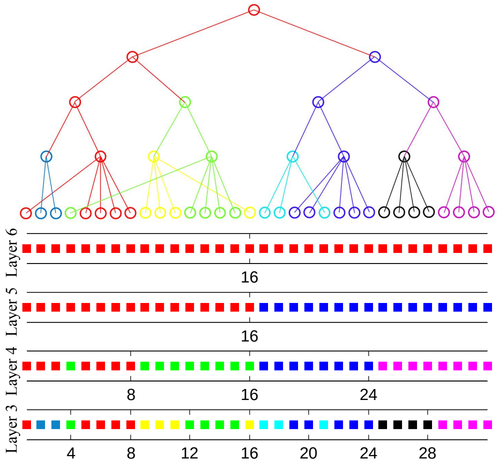

Figure 3: The top 4 layers in the learned task tree ($H = 6$) with two representations for the synthetic data when $H^* = 6$.

the task grouping and the hierarchical approaches and it can learn multi-level task groups, which is similar to the task tree structure, hence, it outperforms both the hierarchical MTL models and the task grouping methods when the height of the task tree is high, e.g., $H^* = 6$ or 7. Among all those methods, the proposed TAT model has the best performance under all the settings. In Table 2, we record the best H in the TAT model selected based on the candidate set. From the results, we see that the estimated H's perfectly match the ground truths, which implies that the proposed TAT model can well capture the tree structure. This also explains why the TAT model can consistently achieve good performance. The left figure in Fig. 4 also reports the sensitivity analysis of the TAT model with respect to H under different settings where each line corresponds to a setting with the ground truth H^* depicted in the legend and the best selected H's are denoted by solid markers. From the results, we can find that the performance is not very sensitive to the number of layers on this synthetic data.

Moreover, since the learned component matrices can be obtained from the TAT model, we can obtain the learned task tree based on

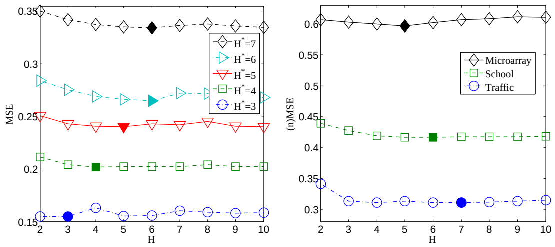

Figure 4: Sensitivity analysis of the TAT model w.r.t. H. Left figure: Results on synthetic data where $H^* = 3, 4, 5, 6, 7$ according to Table 2. Right figure: Results on the microarray, school and traffic datasets.

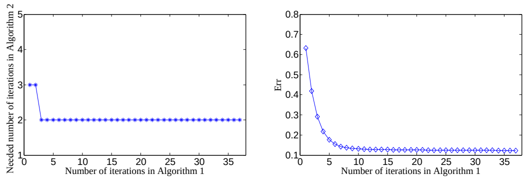

Figure 5: Left figure: The number of iterations required in Algorithm 2. Right figure: The change of Err when varying the number of iterations in Algorithm 1.

the component matrices. Due to the limited precision in the numerical computation, the component parameters for tasks can not be exactly identical and we use the normalized cut algorithm [22] to discover the tree structure layer by layer where in each layer, tasks from a group are assumed to belong to an internal node. Fig. 3 shows the top 4 layers of the learned task tree when $H^* = 6$. We provide two representations for the learned task tree. The first one is the conventional tree representation and the second one is a layerwise representation where the nodes in each layer denote the component parameters in the corresponding component matrix and they have the same color if they belong to one group or equivalently share the same parent node. By comparing with the true task tree at the right side of Fig. 2, we can find that the TAT model has good recovery for the binary tree in the top 4 layers. The left figure in Fig. 5 plots the number of iterations required by Algorithm 2 in each iteration of Algorithm 1, while the right figure shows the change of the relative estimation error (Err), which is defined as Err $= \|\hat{\mathbf{W}} - \mathbf{W}^*\|_F / \|\mathbf{W}^*\|_F$, by varying the number of iterations in Algorithm 1. From the results, we find that both algorithms converge in a fast rate.

6.2 Results on Microarray Data

In this section, we conduct experiments on the microarray data [23],[5] a benchmark dataset for MTL. The microarray data is

[5] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC545783/

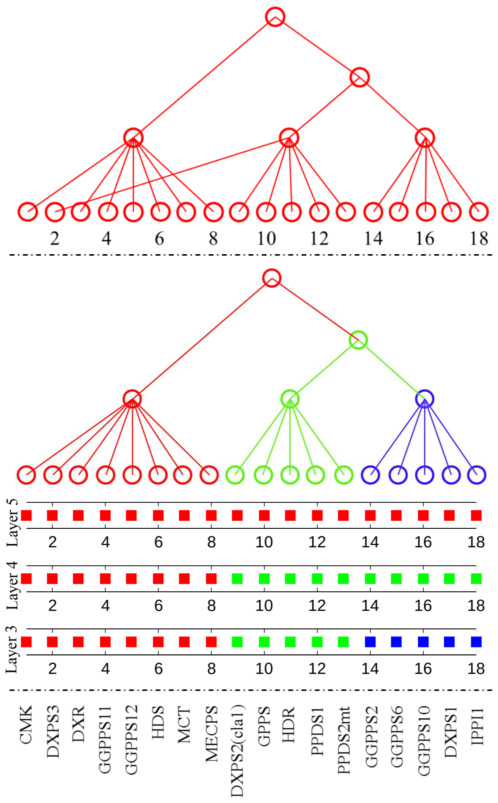

Figure 6: The visualization of the task tree on the microarray data. Top: tree obtained from the hierarchical clustering algorithm. Middle: the learned task tree obtained from the TAT model with two representations. Bottom: the names of the genes.

a gene expression data set related to isoprenoid biosynthesis in plant organism. One major challenge in this problem is to find the cross-talks between two isoprenoid pathways: the mevalonate pathway and the plastidial pathway. The tasks are regression problems where the data features are the expression levels of 21 genes in the mevalonate pathway and the labels are the expression levels of 18 genes in the plastidial pathway, resulting in 18 tasks. There are 118 samples with each feature log-transformed and normalized to have mean 0 and variance 1. According to [23], the genes exhibit tree structure, making it suitable for learning the task tree. We randomly select 60%, 20%, and 20% samples for training, testing, and validation respectively. All the experiments are repeated for 10 times and the average MSE's of different models are reported in Table 3. From Table 3, we see that the TAT model achieves the best performance over all the methods in comparison. We plot the change of the performance of the TAT model with respect to H in the right figure of Fig. 4 and find that the TAT model achieves the best performance when setting H to be 5.

Moreover, we show the top 3 layers of the learned task tree in Fig. 6 for the case that $H = 5$. Since there is no ground truth for the tree structure among tasks, we compare with a tree structure learned from a hierarchical clustering method, which first uses the k-means clustering method to cluster different tasks into 3 clusters and then groups to 2 clusters based on the clustering results in the first step. From the results shown in Fig. 6 where the hierarchical clustering result is at the top and the task tree is in the middle, we see that the learned task tree by the TAT model is very similar to the tree obtained from the hierarchical clustering method, which in some sense demonstrates the ability of the TAT method to find the underlying tree structure among tasks.

6.3 Results on School and Traffic Data

In this section, we experiment on the school and traffic data. The school data[6] is a data set with the exam scores of 15,362 students from 139 secondary schools, where each student is described by 27 attributes. Each learning task is a regression problem to predict the students' exam scores at a school, leading to 139 regression tasks. We randomly select 30%, 50% and 20% samples for training, testing, and validation separately.

The traffic data [11] is to help understand the casual relationships from the entries to the exits of vehicle flows, which is an important problem in traffic systems. This dataset is collected from 272 sensors placed in a highway network, where 136 sensors are placed in the exits of the highways and the others are in the entries. Each exit corresponds to one task and the information collected in entries is considered as the data matrix shared by all the tasks. In each task, there are 384 data samples. We randomly select 20%, 60% and 20% samples for training, testing and validation separately.

All the experiments are repeated for 10 times and the average normalized MSE's (nMSE), i.e. MSE$/\frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i}\|\mathbf{y}_i^*\|_2^2$, of different models are reported in Table 3. Similar to the microarray data, the TAT method has the best performance on both datasets. The best selected H's on the two dataset are 6 and 7, respectively, which are shown in the right figure of Fig. 4.

Table 3: The average (n)MSE's of different methods over 10 random splits on the microarray, school and traffic data.

Model	Microarray	School	Traffic
MTFL	0.6342±0.0913	0.4393±0.0055	0.3523±0.0117
Dirty	0.6141±0.1104	0.4445±0.0051	0.3299±0.0077
Cascade	0.6112±0.0866	0.4366±0.0058	0.3228±0.0070
CMTL	0.6127±0.0887	0.4374±0.0066	0.3367±0.0096
GOMTL	0.6121±0.0877	0.6466±0.0444	0.3258±0.0052
MeTaG	0.6088±0.0881	0.4227±0.0049	0.3116±0.0068
TAT	**0.5966±0.0784**	**0.4169±0.0032**	**0.3107±0.0069**

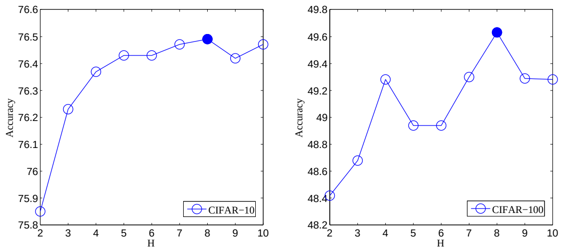

Figure 7: Sensitivity analysis of the TAT model on the CIFAR-10 and CIFAR-100 datasets w.r.t. H. The best selected H's are denoted by solid markers.

6.4 Results on CIFAR Data

In this section, two popular object recognition databases, the CIFAR-10 and CIFAR-100 datasets,[7] are used in our experiments. Each dataset consists of 50,000 color images with size 32 × 32 for training and 10,000 images for testing. The CIFAR-10 data contains 10 classes and in the CIFAR-100 data, there are 100 classes. Each class in those two datasets corresponds to a task and all the tasks share the same training images. By following [5], we use the nonlinear features derived from the k-means algorithm instead of the raw pixels, where $d = 6401$. The GOMTL model fails to work on those two datasets, since the Kronecker product that it requires between two big matrices related to the feature dimensionality is

[6] http://www.cs.ucl.ac.uk/staff/A.Argyriou/code/

[7] http://www.cs.toronto.edu/~kriz/cifar.html

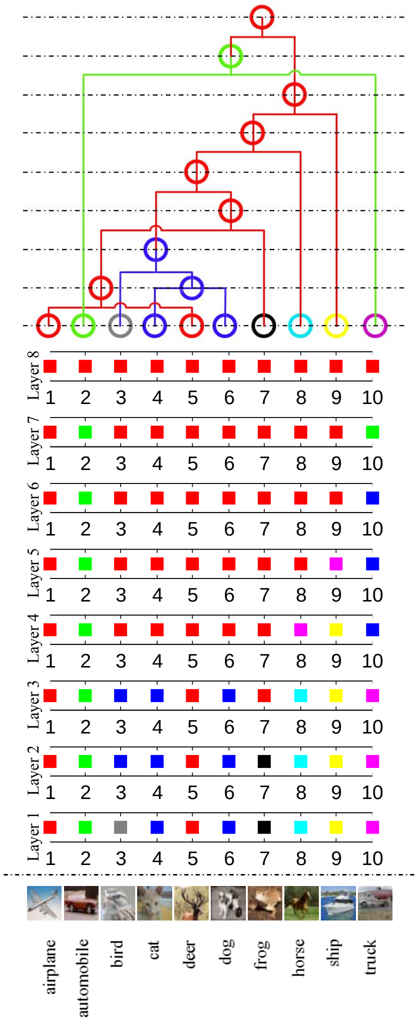

Figure 8: **Two representations of the task tree on the CIFAR-10 data.**

computationally prohibited and hence we do not include it in the comparison. The classification accuracies of different models are shown in Table 4. From the results, we can see that the TAT method has the best performance on those two datasets. Moreover, Fig. 7 shows how the TAT model performs when varying H and we can find that the best H's in those two datasets are both equal to 8.

Moreover, the best learned task tree, which has 8 layers, on the CIFAR-10 data is shown in Fig. 8. From Fig. 8, we can find some interesting results. For example, tasks 'automobile' and 'truck' are identified to belong to a group at the 7th layer while the other tasks belong to another one. Tasks 'cat' and 'dog' always belong to the same group in the task tree and all the tasks related to animals (i.e., bird, cat, deer, dog, frog, and horse) are discovered to belong to a group at the 5th layer and above. Moreover, almost all the tasks related to manufactures (e.g. automobile, ship, and truck) are always identified to belong to different groups below the 5th layer due to their different shapes. All the above findings are intuitively reasonable and this shows that the TAT model can find meaningful task relations contained in the data.

Table 4: **The accuracies of different methods on the CIFAR-10 and CIFAR-100 datasets.**

Data	MTFL	Dirty	Cascade	CMTL	MeTaG	TAT
CIFAR-10	71.15	72.56	74.76	74.49	75.99	**76.49**
CIFAR-100	39.61	42.45	46.12	46.70	48.62	**49.63**

6.5 Efficiency Testing for Algorithm 3

Table 5: **Comparison on the total running time (in seconds).**

Length	5	10	20	1000	5000	10000
Num. of seq.	1000	1000	1000	10	10	10
CVX	178.8	195.4	220.8	39.7	131.3	303.8
Algorithm 3	0.27	0.69	0.94	0.6	2.9	5.6
Speedup	**662.2**	**283.1**	**234.9**	**66.2**	**45.3**	**54.3**

In this section, we test the efficiency of Algorithm 3, which is a key step in the whole optimization procedure and needs to be executed frequently. We compare Algorithm 3 with the CVX solver [10] since problem (17) is convex. The experimental platform is the Matlab 2013b running on a machine with Intel i7 CPU and 8GB RAM.

We generate random sequences to test Algorithm 3. The sequence in the sequential constraints is of length H and from the previous experiments, we can see that it is usually small, e.g., a constant between 2 and 10. We randomly generate 1000 sequences with length 5, 10, and 20 respectively, and test the algorithms on them. Moreover, we also do some experiments under the setting that the length of the sequence is large and generate 10 sequences with length 1000, 5000, and 10000 respectively. The total running time for the two settings is reported in Table 5, from which we see that the proposed Algorithm 3 is very efficient under all the settings.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel TAT model to learn the underlying tree structure for multi-task learning. We developed an efficient algorithm to solve the non-convex problem in the proposed TAT model and provided theoretical analysis. Experimental results show that the task tree learned by the TAT model can provide deep understanding on the task relations contained in the data.

Currently, the number of layers in the TAT model needs to be predefined. In future work, we are interested in learning the number of layers from data automatically.

Acknowledgments

This work is supported by Natural Science Foundations of China under Grant No. 61305071.

APPENDIX

A. PROOF OF LEMMA 1

Proof. The first statement is obvious. We now adopt the induction technique to prove the second statement.

When $H = 2$ and $\bar{u}_1 \le \bar{u}_2$, we assume the optimal solution is $(\bar{q}_1^*, \bar{q}_2^*)$, where $\bar{q}_1^* \ge \bar{q}_2^*$. If $\bar{q}_1^* > \bar{q}_2^*$, there must exist a $\check{q}$ that $\bar{q}_1^* > \check{q} > \bar{q}_2^*$ and $\bar{u}_1 \le \check{q} \le \bar{u}_2$. Otherwise, if $\bar{u}_1 \le \bar{u}_2 < \check{q} < \bar{q}_1^*$, we can immediately get $\bar{q}_2^* = \bar{u}_2$, and then $(\bar{q}_2^*, \bar{q}_2^*)$ is better than $(\bar{q}_1^*, \bar{q}_2^*)$, which contradicts the fact that $(\bar{q}_1^*, \bar{q}_2^*)$ is the optimal solution. The case that $\bar{q}_2^* < \check{q} < \bar{u}_1 \le \bar{u}_2$ can be proved similarly. Now we have $\bar{q}_1^* > \check{q} > \bar{q}_2^*$ and $\bar{u}_1 \le \check{q} \le \bar{u}_2$. Assume $\check{q} = \frac{\bar{u}_1+\bar{u}_2}{2}$, we can immediately obtain that $(\check{q}, \check{q})$ is better than $(\bar{q}_1^*, \bar{q}_2^*)$, which again contradicts the fact that $(\bar{q}_1^*, \bar{q}_2^*)$ is the optimal solution. Therefore, we must have $\bar{q}_1^* = \bar{q}_2^* = \frac{\bar{u}_1+\bar{u}_2}{2}$.

Then we assume that the statement holds for any $H \le n-1$. We will show that when $H = n$, the statement also holds. Actually, given $H = n$ and $\bar{u}_1 \le \cdots \le \bar{u}_n$, the optimal solution must have the form $(\check{q}, \cdots, \check{q})|_{n-1} \bowtie \bar{q}_n^*$, i.e. $(\check{q}, \cdots, \check{q}, \bar{q}_n^*)$, where $\check{q} \ge \bar{q}_n^*$. Otherwise, suppose the optimal solution is denoted by

$(\bar{q}'_1, \bar{q}'_2, \cdots, \bar{q}'_n)$ with at least one equality dissatisfied in inequalities $\bar{q}'_1 \geq \bar{q}'_2 \geq \cdots \geq \bar{q}'_n$. Then we can immediately obtain a contradiction that the sequence $(\check{q}', \cdots, \check{q}')|_{n-1} \bowtie \bar{q}'_n$ is better than $(\bar{q}'_1, \bar{q}'_2, \cdots, \bar{q}'_n)$ where $(\check{q}', \cdots, \check{q}')|_{n-1}$ is the optimal solution of the problem of size $H = n - 1$ corresponding to the sequence $(\bar{u}_1, \ldots, \bar{u}_{n-1})$. Similarly, we can get that the optimal solution have the form $\bar{q}^*_1 \bowtie (\check{q}, \cdots, \check{q})|_{n-1}$, i.e. $(\bar{q}^*_1, \check{q}, \cdots, \check{q})$, where $\bar{q}^*_1 \geq \check{q}$. Combing those results we complete the proof. ■

B. PROOF OF LEMMA 2

Proof. We first prove it when $u^* \geq u' \geq b_1$. Given any $H = n$ and the sequence $(\bar{u}_1, \cdots, \bar{u}_n)$, we consider the feasible sequence $(b_1, \cdots, b_n)$ for problem (17), where $b_1 \geq \cdots \geq b_n$. Then, we can obtain that the sequence $(b_1, \cdots, b_1)|_n$ is not worse than $(b_1, \cdots, b_n)$, because if $(b_1, \cdots, b_1)|_n$ is worse, there must exist a sequence $(\breve{b}_2, \cdots, \breve{b}_n)$, where $u^* > \breve{b}_2 \geq \cdots \geq \breve{b}_n$, such that the optimal solution for the sub-sequence $(\bar{u}_2, \cdots, \bar{u}_n)$ is $(\breve{b}_2, \cdots, \breve{b}_n)$, and in that case $(u^*, \breve{b}_2, \cdots, \breve{b}_n)$ is better than $(u^*, \cdots, u^*)|_n$, which contradicts with the fact that $(u^*, \cdots, u^*)|_n$ is the optimal solution. Therefore $(b_1, \cdots, b_1)|_n$ is better than $(b_1, \cdots, b_n)$. Furthermore, since $u^* \geq u' \geq b_1$, it is easy to see that $(u', \cdots, u')|_n$ is not worse than $(b_1, \cdots, b_1)$ due to the convexity of the function $f(x) = \sum_h (x - \bar{u}_h)^2$. So we complete the proof when $u^* \geq u' \geq b_1$. The case that $b_H \leq u' \leq u^*$ can be proved similarly and we finish the proof. ■

C. PROOF OF THEOREM 1

Proof. The case that $\dot{u}^* \geq \ddot{u}^*$ is obvious. Then we prove the case that $\dot{u}^* < \ddot{u}^*$. In this case, we denote the optimal solution for the concatenated sequence by $(\bar{q}^*_1, \cdots, \bar{q}^*_l, \bar{q}^*_{l+1}, \cdots, \bar{q}^*_n)$, where $\bar{q}^*_1 \geq \cdots \geq \bar{q}^*_l \geq \bar{q}^*_{l+1} \geq \cdots \geq \bar{q}^*_n$. Then it is easy to show that $\bar{q}^*_l \geq \dot{u}^*$, because if $\bar{q}^*_l < \dot{u}^*$, substituting the sub-sequence $(\bar{q}^*_1, \cdots, \bar{q}^*_l)$ with $(\dot{u}^*, \cdots, \dot{u}^*)|_l$ in $(\bar{q}^*_1, \cdots, \bar{q}^*_l, \bar{q}^*_{l+1}, \cdots, \bar{q}^*_n)$ will lead to a better feasible solution, which makes a contradiction. Similarly, we can show that $\bar{q}^*_{l+1} \leq \ddot{u}^*$. Then based on Lemma 2, substituting the two sub-sequences $(\bar{q}^*_1, \cdots, \bar{q}^*_l)$ and $(\bar{q}^*_{l+1}, \cdots, \bar{q}^*_n)$ with $(\bar{q}^*_l, \cdots, \bar{q}^*_l)|_l$ and $(\bar{q}^*_{l+1}, \cdots, \bar{q}^*_{l+1})|_{n-l}$ respectively will generate a new solution that is not worse than the previous one. Note that $\bar{q}^*_l \geq \dot{u}^*$, $\bar{q}^*_{l+1} \leq \ddot{u}^*$, $\dot{u}^* < \ddot{u}^*$ and $\bar{q}^*_l \geq \bar{q}^*_{l+1}$. Then the optimal solution is achieved when $\bar{q}^*_l = \bar{q}^*_{l+1}$ due to the convexity of the objective function, making the optimal solution have the form $(u^*, \cdots, u^*)|_n$. Plugging the form into problem (17), we get $u^* = \frac{\sum_{i=1}^n \bar{u}_i}{n}$, in which we reach the conclusion. ■

D. PROOF OF THEOREM 2

Proof. In Algorithm 3, step 1 splits the initial sequence $(\bar{u}_1, \cdots, \bar{u}_H)$ into non-decreasing sub-sequences. According to Lemma 1, the solutions for those non-decreasing sub-sequences take the form that the entries in the solution are identical. Then, steps 2-14 concatenate the solutions of these sub-sequences according to Theorem 1 iteratively. According to Theorem 1, the global optimality can be guaranteed for any concatenation operation. So Algorithm 3 can find the optimal solution in step 15 for problem (17). ■

References

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

[2] R. Caruana. Multitask learning. *MLJ*, 28(1):41–75, 1997.

[3] J. Chen, L. Tang, J. Liu, and J. Ye. A convex formulation for learning shared structures from multiple tasks. In *ICML*, 2009.

[4] J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *KDD*, 2011.

[5] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[6] T. Evgeniou, C. A. Micchelli, M. Pontil, and J. Shawe-Taylor. Learning multiple tasks with kernel methods. *JMLR*, 6(4), 2005.

[7] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *KDD*, 2004.

[8] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *ICML*, 2013.

[9] N. Görnitz, C. K. Widmer, G. Zeller, A. Kahles, G. Rätsch, and S. Sonnenburg. Hierarchical multitask structured output learning for large-scale sequence segmentation. In *NIPS*, 2011.

[10] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, 2014.

[11] L. Han, G. Song, G. Cong, and K. Xie. Overlapping decomposition for causal graphical modeling. In *KDD*, 2012.

[12] L. Han and Y. Zhang. Learning multi-level task groups in multi-task learning. In *AAAI*, 2015.

[13] L. Han, Y. Zhang, G. Song, and K. Xie. Encoding tree sparsity in multi-task learning: A probabilistic framework. In *AAAI*, 2014.

[14] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *NIPS*, 2008.

[15] A. Jalali, P. Ravikumar, S. Sanghavi, and C. Ruan. A dirty model for multi-task learning. In *NIPS*, 2010.

[16] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. In *ICML*, 2011.

[17] S. Kim and E. P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010.

[18] A. Kumar and H. Daume III. Learning task grouping and overlap in multi-task learning. In *ICML*, 2012.

[19] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *UAI*, 2009.

[20] K. Lounici, M. Pontil, A. B. Tsybakov, and S. van de Geer. Taking advantage of sparsity in multi-task learning. In *COLT*, 2009.

[21] A. C. Lozano and G. Swirszcz. Multi-level lasso for sparse multi-task regression. In *ICML*, 2012.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22(8):888–905, 2000.

[23] A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelić, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem, and P. Bühlmann. Sparse graphical Gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biol*, 5(11):R92, 2004.

[24] Y. Zhang. Heterogeneous-neighborhood-based multi-task local learning algorithms. In *NIPS*, 2013.

[25] Y. Zhang and J. G. Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *NIPS*, 2010.

[26] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *UAI*, 2010.

[27] Y. Zhang, D.-Y. Yeung, and Q. Xu. Probabilistic multi-task feature selection. In *NIPS*, 2010.

[28] A. Zweig and D. Weinshall. Hierarchical regularization cascade for joint learning. In *ICML*, 2013.